

Capitolo 5 – Gestione Avanzata dei Container

In questo capitolo vedremo:

- Gestione avanzata dei container
 - Volumi Docker
 - Reti Docker
 - Variabili d'ambiente
 - Limiti di risorse (CPU e RAM)
 - Policy di riavvio
-

5.1 Gestione dei container

Elencare i container

Container attivi:

```
docker ps
```

Tutti i container (anche fermati):

```
docker ps -a
```

Avviare un container fermo

```
docker start nome_container
```

Fermare un container

```
docker stop nome_container
```

Riavviare un container

```
docker restart nome_container
```

Eliminare un container

```
docker rm nome_container
```

Forzare la rimozione:

```
docker rm -f nome_container
```

5.2 Log dei container

Visualizzare i log:

```
docker logs nome_container
```

Seguire i log in tempo reale:

```
docker logs -f nome_container
```

5.3 Accesso alla shell di un container

Entrare in un container in esecuzione:

```
docker exec -it nome_container /bin/bash
```

Se bash non è disponibile:

```
docker exec -it nome_container /bin/sh
```

5.4 Variabili d'ambiente

Passare variabili al container:

```
docker run -d -e MYSQL_ROOT_PASSWORD=secret mysql
```

Oppure tramite file `.env`:

```
docker run --env-file .env nginx
```

5.5 Volumi Docker

I volumi permettono di salvare dati persistenti fuori dal container.

Creare un volume

```
docker volume create mio_volume
```

Elencare i volumi:

```
docker volume ls
```

Utilizzare un volume

```
docker run -d \  
  -v mio_volume:/var/lib/mysql \  
  mysql
```

In questo modo i dati non vengono persi se il container viene eliminato.

Rimuovere un volume

```
docker volume rm mio_volume
```

5.6 Bind Mount (cartella host)

Montare una cartella del sistema host:

```
docker run -d \  
  -v /home/user/dati:/app/dati \  
  nginx
```

Differenza:

- Volume → gestito da Docker
 - Bind mount → directory reale del sistema host
-

5.7 Reti Docker

Elencare le reti

```
docker network ls
```

Creare una rete personalizzata

```
docker network create mia_rete
```

Avviare container nella stessa rete

```
docker run -d --name app --network mia_rete nginx
docker run -d --name db --network mia_rete mysql
```

I container nella stessa rete possono comunicare usando il nome del container come hostname.

5.8 Limiti di risorse

Limitare la memoria:

```
docker run -d --memory="512m" nginx
```

Limitare la CPU:

```
docker run -d --cpus="1.0" nginx
```

Questo evita che un container utilizzi tutte le risorse del sistema.

5.9 Policy di riavvio

Riavviare automaticamente il container:

```
docker run -d --restart unless-stopped nginx
```

Opzioni disponibili:

- no
 - always
 - on-failure
 - unless-stopped
-

5.10 Ispezione e statistiche

Dettagli completi di un container:

```
docker inspect nome_container
```

Utilizzo risorse in tempo reale:

```
docker stats
```

Best Practice

- Usare volumi per dati persistenti
 - Limitare CPU e RAM in ambienti di produzione
 - Utilizzare reti personalizzate per isolamento
 - Impostare policy di riavvio per servizi critici
 - Monitorare log e risorse regolarmente
-

Conclusione

Ora sai:

- Gestire container in modo avanzato
- Usare volumi e bind mount
- Configurare reti personalizzate
- Limitare risorse
- Configurare riavvii automatici

Nel prossimo capitolo vedremo Docker Compose per gestire applicazioni multi-container.

Revision #1

Created 2026-02-27 16:38:05 UTC by Pe

Updated 2026-02-27 16:38:24 UTC by Pe